# An Algebraic Framework for Urgency

Sébastien Bornot and Joseph Sifakis
Sebastien.Bornot@imag.fr Joseph.Sifakis@imag.fr

Verimag, 2 rue Vignate, 38610 Gières, France

## 1 Introduction

Timed formalisms are extensions of untimed ones by adding *clocks*, real-valued variables that can be tested and modified at transitions. Clocks measure the time elapsed at states when some implicitly or explicitly given *time progress conditions* are satisfied. Timed automata, timed process algebras and timed Petri nets can be considered as timed formalisms.

The semantics of timed formalisms can be defined by means of transition systems that perform time steps or (timeless) transitions. Clearly, such transition systems ought to be well-timed in the sense that it is possible for time to progress forever. It is recognized that the compositional description of timed systems that satisfy even weak well-timedness requirements, is a non trivial problem. An inherent difficulty is that usually, the semantics of operators compose separately time steps and transitions by preserving urgency: time can progress in a system by some amount if all its components respect their time progress constraints. This leads to semantics based on a nice "orthogonality principle" between time progress and discrete state changes. Parallel composition and other operators have been defined according to this principle, for timed process algebras and hybrid automata. However, composing independently time steps and transitions may easily introduce timelocks. It is questionable if the application of a strong synchronization rule for time progress is always appropriate. For instance, if two systems are in states from which they will never synchronize, it may be desirable not to further constrain time progress by the strong synchronization rule.

In several papers ([SY96,BS98,BST97]) we have studied compositional description methods that are based on "flexible" composition rules that relax urgency constraints so as to preserve a weak well-timedness property that we call *time reactivity*. The latter means that if no discrete transition can be executed from a state then time can progress. Contrary to other stronger properties, time reactivity is very easy to satisfy by relating directly time progress conditions and enabling conditions of discrete transitions. We have proposed a simple sub-class of timed automata, called *timed automata with deadlines* that are time reactive and we have shown how choice and parallel composition operators that preserve time reactivity can be defined. In this paper, we present a unified algebraic framework that encompasses the already presented results and provides laws for choice and parallel composition on timed systems, modulo strong bisimulation. The algebraic framework is characterized by the following.

- Timed systems are obtained as the composition of *timed actions* by using operators. A timed action is a discrete transition, labeled with an action name, a guard, a deadline and a jump. Guards and deadlines are predicates on clocks characterizing respectively, the states at which the action is enabled and the states at which the action becomes urgent (time progress stops). We require that the deadline implies the corresponding guard which guarantees time reactivity. The jumps are functions that specify clock assignments when the action is executed.
- The operators are timed extensions of untimed operators. They preserve both time reactivity and *activity* of components. The latter is the property meaning that if some action can be executed after waiting some amount of time in a component, then some action of the composed system can be executed after waiting some (not necessarily the same) amount of time.
  We propose timed extensions of choice and parallel composition operators that are associative and commutative and are related by an expansion theorem. Choice operators are parameterized by an order relation on actions that is proven to be useful, in particular to define parallel composition with maximal progress.
- In addition to the usual laws for untimed operators, timed operators satisfy specific laws reflecting the structure of timed actions and assumptions about their synchronization. We identify different synchronization modes that take into account the possibility of waiting of the components and study their properties.

The paper is organized as follows. Section 2 presents the basic model, which is essentially automata with clocks, an abstraction of timed automata without the usual restrictions on guards and assignments. Section 3 and section 4 present respectively, basic results on priority choice operators and parallel composition, such as associativity, activity preservation and the expansion theorem. Section 5 describes the algebraic framework. Two examples illustrating its use are given in section 6. We conclude by discussing future work directions and relations to existing work.

## 2 Timed Systems

### 2.1 Background

Let $X$ be a set of real-valued variables called clocks defined on the set of non-negative reals $\mathbf{R}_{\geq \mathbf{0}}$. Clocks will be used as state variables measuring time progress. The set of the valuations of $X$ isomorphic to ${\mathbf{R}_{\geq \mathbf{0}}}^n$ for some $n$, is denoted by $V$. Constant $true$ (resp. $false$) denotes the predicate that is true (resp. false) for any valuation $v \in V$. For any non-negative real $t$, we represent by $v+t$ the valuation obtained from $v$ by increasing by $t$ the values of all the clocks.

**Definition 1.** Left- and right-closure
A predicate $p$ on $X$ is called left-closed if

$$\forall v \;.\; \neg p(v) \Rightarrow \exists \epsilon > 0 \;.\; \forall \epsilon' \leq \epsilon \;.\; \neg p(v + \epsilon')$$

It is called right-closed if it satisfies the previous expression where $p(v+\epsilon')$ is replaced by $p(v-\epsilon')$.

Notice that these two definitions correspond to the usual notions if we consider $p$ as a function of time, where $v$ is a clock valuation.

**Definition 2.** Rising and falling edge
Given a predicate $p$ on clocks $X$, we define the rising edge of $p$, noted $p\uparrow$ by:

$$p\uparrow(v) = p(v) \wedge \exists \epsilon > 0 \;.\; \forall \epsilon' \in (0,\epsilon] \;.\; \neg p(v-\epsilon') \vee \\ \neg p(v) \wedge \exists \epsilon > 0 \;.\; \forall \epsilon' \in (0,\epsilon] \;.\; p(v+\epsilon')$$

The falling edge of $p$, noted $p\downarrow$, is defined by the same formula where $v-\epsilon'$ and $v+\epsilon'$ are exchanged.

**Definition 3.** Modal operators
Given a predicate $p$ on $V$, we define the modal operators $\Diamond_k\ p$ ("eventually $p$ within $k$") and $\diamondsuit_k\ p$ ("once $p$ since $k$"), for $k \in \mathbf{R}_{\geq \mathbf{0}} \cup \{\infty\}$.

$$\Diamond_k\ p\ (v) \text{ iff } \exists t \in \mathbf{R}_{\geq \mathbf{0}}\ 0 \leq t \leq k.\ p(v+t)$$
$$\diamondsuit_k\ p\ (v) \text{ iff } \exists t \in \mathbf{R}_{\geq \mathbf{0}}\ 0 \leq t \leq k.\ \exists v' \in V.\ v = v' + t \wedge p(v')$$

We write $\Diamond p$ and $\diamondsuit\ p$ for $\Diamond_\infty\ p$ and $\diamondsuit_\infty\ p$, respectively, and $\Box p$ and $\boxdot\ p$ for $\neg\Diamond\neg p$ and $\neg\diamondsuit\neg p$, respectively.

Notice that the operators $\Diamond_k$ and $\diamondsuit_k$ are just a notation for existential quantification over time and should not be confused with temporal logic operators. Expressions with modal or edge operators can be reduced to predicates on $X$ whenever quantification over time can be eliminated e.g., when the operators are applied to linear constraints on $X$. For example, $\Diamond(1 \leq x \leq 2)$ is equivalent to $x \leq 2$ and $\Diamond_2(3 \leq x \leq 5)$ is equivalent to $1 \leq x \leq 5$.

### 2.2 Timed Systems

**Definition 4.** Timed systems
A Timed System is:

- An untimed labeled transition system $(S, \rightarrow, A)$ where
  - $S$ is a finite set of *control states*
  - $A$ is a finite vocabulary of *actions*
  - $\rightarrow \subseteq S \times A \times S$ is an untimed transition relation
- A finite set $X$ of clocks.
- A labeling function $h$ mapping *untimed transitions* of $\rightarrow$ into *timed transitions*: $h(s,a,s') = (s,(a,g,d,f),s')$, where
  - $g$ and $d$ are predicates on $X$ called respectively the *guard* and the *deadline* of the transition. We require that $d \Rightarrow g$.
  - $f : V \rightarrow V$ is a *jump.*

According to the above definition, a timed system can be obtained from an untimed one by associating with each action $a$, a *timed action* $b = (a, g, d, f)$.

**Definition 5.** Semantics of timed systems
A *state* of a timed system is a pair $(s, v)$, where $s \in S$ is a control state and $v \in V$. We associate with a timed system a transition relation $\rightarrow \subseteq (S \times V) \times (A \cup \mathbf{R}_{\geq \mathbf{0}}) \times (S \times V)$. Transitions labeled by elements of $A$ are *discrete transitions* while transitions labeled by non-negative reals are *time steps*.

Given $s \in S$, if $\{(s, a_i, s_i)\}_{i \in I}$ is the set of all the untimed transitions issued from $s$ and $h(s, a_i, s_i) = (s, (a_i, g_i, d_i, f_i), s_i)$ then:

- $\forall i \in I \ \forall v \in \mathbf{R}_{\geq \mathbf{0}} \ . \ (s, v) \stackrel{a_i}{\rightarrow} (s_i, f_i(v))$ if $g_i(v)$.
- $(s, v) \stackrel{t}{\rightarrow} (s, v + t)$ if $\forall t' < t \ . \ c_s(v + t')$ where $c_s = \neg \bigvee_{i \in I} d_i$.
  Thus, time can progress at control state $s$, as long as no deadline of a transition from $s$ becomes true. We call $c_s$ the *time progress condition* associated with the control state $s$.

We consider timed systems such that for any control state $s$ if the time progress condition $c_s$ is right-closed then its falling edge is implied by the guard of a transition from $s$.

The condition $d \Rightarrow g$ guarantees that if time cannot progress at some state, then some action is enabled from this state. Restriction to systems with an enabled transition when a time progress condition is right-closed permits to avoid deadlock situations in the case of transitions $(s, (a, g, d, f), s')$ such that $g = d$. For instance, consider the case where $d = g = x > 2$, implying the time progress condition $x \leq 2$, which is right-closed. Then, if $x$ is initially 2, time cannot progress by any time $t$, according to definition 5. The guard $g$ is not satisfied either. Thus, the system is deadlocked. The assumptions above implies the property of *time reactivity*, that is, time can progress at any state unless some untimed transition is enabled. Throughout the paper this will be ensured by only considering timed systems with left-closed guards and deadlines, and operators that preserve time reactivity for these systems.

The semantics of a timed system is its associated transition relation modulo strong bisimulation, usually called timed bisimulation.

**Definition 6.** Timed bisimulation
Two states $(s_1, v_1)$ and $(s_2, v_2)$ are timed bisimilar if there exists a symmetric relation $\rho \in (S \times V)^2$ such that
$((s_1, v_1), (s_2, v_2)) \in \rho$ and $(s_1, v_1) \stackrel{\lambda}{\rightarrow} (s_1', v_1')$ for some $\lambda \in A \cup \mathbf{R}_{\geq \mathbf{0}}$ implies $(s_2, v_2) \stackrel{\lambda}{\rightarrow} (s_2', v_2')$ for some $(s_2', v_2')$ where $((s_1', v_1'), (s_2', v_2')) \in \rho$.

We introduced timed systems as an abstraction of TAD [BS98] obtained by relaxation of usual syntactical restrictions ensuring decidability. TAD can be considered as a sub-class of time reactive timed automata [HNSY94] where invariants associated with control states are replaced by deadlines.

The simplest timed system is a single transition labeled with the timed action $(a, g, d, f)$. The guard $g$ characterizes the set of states from which the timed

transition is possible, while the deadline $d$ characterizes the subset of these states where the timed transition is enforced by stopping time progress. The relative position of $d$ within $g$ determines the urgency of the action. For a given $g$, the corresponding $d$ may take two extreme values: $d = g$ meaning that the action is *eager*, and $d = false$, meaning that the action is *lazy*. A particularly interesting case is the one of a *delayable* action where $d = g\downarrow$ is the falling edge of a right-closed guard $g$ (cannot be disabled without enforcing the action). The differences between urgency types are illustrated in figure 1.

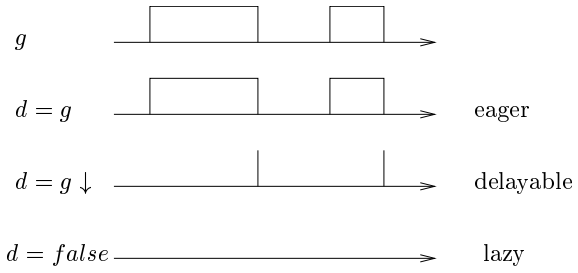



**Fig. 1.** Using deadlines to specify urgency.

## 3 Choice Operators

### 3.1 The Algebra of Regular Processes

In this paragraph, we summarize basic results about the algebraic notation that will be used throughout the paper [Mil89].

Consider the language of terms $P(A)$ built from a constant $Nil$ and a set of variables $VAR$ by using prefixing by actions of a vocabulary $A$, choice and recursion.

$$p ::= \ Nil \ \mid Z \in VAR \ \mid \ a.p \ \mid \ p + p \ \mid \ rec\ Z.p$$

We adopt the usual notion for free and bounded variables and guarded definition.

The terms of this language represent labeled transition systems on $A$. The transition relation is the union of the least relations $\{\stackrel{a}{\rightarrow}\}_{a\in A}$ such that

$$\begin{array}{l} a.p \stackrel{a}{\rightarrow} p \\ p_1 \stackrel{a}{\rightarrow} {p_1}' \ \text{ implies } \ p_1 + p_2 \stackrel{a}{\rightarrow} {p_1}' \ \text{ and } \ p_2 + p_1 \stackrel{a}{\rightarrow} {p_1}' \\ p[rec\ Z.p/Z] \stackrel{a}{\rightarrow} p' \ \text{ implies } \ rec\ Z.p \stackrel{a}{\rightarrow} p' \end{array}$$

where $p[rec\ Z.p/Z]$ is the term obtained by substituting in $p$ the free occurrences of variable $Z$ by $rec\ Z.p$.

The algebra of regular processes is the algebra of terms defined above modulo the congruence induced by the following inference system, called strong congruence.

$$\begin{array}{ll} p_1 + p_2 = p_2 + p_1 & commutativity \\ (p_1 + p_2) + p_3 = p_1 + (p_2 + p_3) & associativity \\ p + p = p & idempotence \\ p + Nil = p & neutral\ element \\ p[rec\ Z.p/Z] = rec\ Z.p & fixpoint \end{array}$$

if $p$ is well guarded $\qquad p[p'/Z] = p'$ implies $rec\ Z.p = p'$

Strong congruence agrees with strong bisimulation on labeled transition systems in the sense that strongly congruent terms represent strongly bisimilar labeled transition systems.

A consequence of these results is that given $(S, \rightarrow, A)$ a labeled transition system, it can be uniquely characterized modulo strong congruence by a set of equations in bijection with the control states. If for some control state $s$ the set of the exiting transitions is $\{(s, a_i, s_i)\}_{i \in I}$, then the corresponding equation is $Z_s = \sum_{i \in I} a_i.Z_{s_i}$ where $\sum_{i \in I} a_i.Z_{s_i}$ is taken to be $Nil$ if $I = \emptyset$.

### 3.2 Extension to Timed Systems

In the sequel, we consider timed systems as labeled transition systems on a set of timed actions $B = \{b_i\}_{i \in I}$ where $b_i = (a_i, g_i, d_i, f_i)$ for some action $a_i \in A$, guard $g_i$, deadline $d_i$, jump $f_i$. Equality of timed actions means equality of the corresponding components that is $b_1 = b_2$ if $a_1 = a_2$, $g_1 = g_2$, $d_1 = d_2$, $f_1 = f_2$.

We use terms of $P(B)$, regular processes on the vocabulary $B$, to represent timed systems $(S, \rightarrow, A, X, h)$. To simplify notation, we use $s \in S$ to denote the corresponding variable $Z_s$ and write $s = \sum_{i \in I} b_i.s_i$ for the characteristic equation of $s$, where $\{(s, a_i, s_i)\}_{i \in I}$ is the set of the untimed transitions issued from $s$ such that $h(s, a_i, s_i) = b_i$.

Notice that strong congruence in this context is strong bisimilarity of the control structure of timed systems. Another equivalence for the comparison of terms (control states) is obtained by extending timed bisimulation.

**Definition 7.** Timed bisimulation on terms
Two terms $s_1$, $s_2$ are said to be timed bisimilar if for any valuation $v \in V$, the states $(s_1, v)$ and $(s_2, v)$ are timed bisimilar.

As timed bisimulation on terms admits no simple syntactic characterization (see figure 2), we prefer working with strong congruence.

The following proposition can be shown by induction on the structure of the terms. It guarantees that strong congruence on timed systems is compatible with timed bisimulation.

**Proposition 8.** *If two terms $s_1$, $s_2$ are strongly congruent then they are timed bisimilar.*

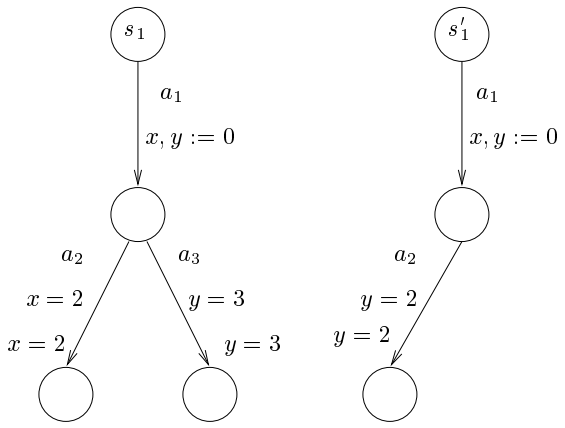



**Fig. 2.** $s_1$ and $s'_1$ are timed bisimilar but not strongly congruent

Throughout the paper, we represent timed systems by well-guarded terms of $P(B)$, of the form $\sum_{i \in I} b_i.s_i$ where $I$ is finite, modulo strong congruence.

### 3.3 Priority Choice

#### Motivation

When from a given state, several timed actions are enabled, it is often useful to reduce non-determinism by using priorities on actions. Intuitively, applying priority implies preventing low priority actions from being executed when higher priority actions are enabled. This amounts to taking the non-deterministic choice between the considered actions by adequately restricting the guards of the actions of lower priority.

Consider, for example, two timed transitions $(s, (a_i, g_i, d_i, f_i), s_i)$, for $i = 1, 2$, with a common source control state $s$. If action $a_1$ has lower priority than $a_2$, in the resulting timed system the transition labeled by $a_2$ does not change while the transition labeled by $a_1$ would be of the form $(s, (a_1, g'_1, d'_1, r_1), s_1)$, where $g'_1 \Rightarrow g_1$ and $d'_1 = d_1 \wedge g'_1$.

For untimed systems, $g'_1$ is usually taken to be $g_1 \wedge \neg g_2$, which means that whenever $a_1$ and $a_2$ are simultaneously enabled, $a_1$ is disabled in the prioritized choice. However, for timed systems other ways to define $g'_1$ are possible. One may want to prevent action $a_1$ from being executed if it is established that $a_2$ will be eventually executed within a given delay. We can take $g'_1 = g_1 \wedge \neg \Diamond_k g_2$ or even $g'_1 = g_1 \wedge \Box \neg g_2$. In the former case, $a_1$ gives priority up to $a_2$ if $a_2$ is eventually enabled within $k$ time units. In the latter case, $a_1$ is enabled only if $a_2$ is disabled forever.

**Definition and Results**

For timed systems, priorities between actions can be parameterized by the amount of time actions of lower priority leave precedence to actions of higher priority. This motivates the following definition.

**Definition 9.** Priority order
A priority order is a relation $\prec \subseteq A \times (\mathbf{R}_{\geq \mathbf{0}} \cup \{\infty\}) \times A$ satisfying the following properties, where $a_1 \prec_k a_2$ stands for $(a_1, k, a_2) \in \prec$,

- $\prec_k$ is a strict partial order for all $k \in \mathbf{R}_{\geq \mathbf{0}} \cup \{\infty\}$
- $a_1 \prec_k a_2$ implies $\forall k' < k.\ a_1 \prec_{k'} a_2$
- $a_1 \prec_k a_2 \wedge a_2 \prec_l a_3$ implies $a_1 \prec_{k+l} a_3$

**Property :** The relation $a_1 \lessdot a_2 = \exists k\ a_1 \prec_k a_2$ is a strict partial order.

*Proof.* $\lessdot$ is irreflexive and transitive by definition. It is antisymmetric : if $a_1 \prec_k a_2$ then for every $k' \leq k$, $a_1 \prec_{k'} a_2$ and since $\prec_0$ is antisymmetric, $a_2 \prec_0 a_1$ does not hold; this implies that for any $k' \in \mathbf{R}_{\geq \mathbf{0}} \cup \{\infty\}$, $a_2 \prec_{k'} a_1$ does not hold. □

**Definition 10.** Binary priority choice
For a given priority order $\prec$, let $B_I = \{b_i\}_{i \in I}$ and $B_J = \{b_j\}_{j \in J}$ denote sets of timed actions with $b_l = (a_l, g_l, d_l, f_l)$, for $l \in I \cup J$. The operator $\widehat{+}$ is a binary operator on timed system defined by
$(\sum_{i \in I} b_i.s_i) \widehat{+} (\sum_{j \in J} b_j.s_j) = (\sum_{i \in I} (b_i \backslash B_J).s_i) + (\sum_{j \in J} (b_j \backslash B_I).s_j)$ with

$$
\begin{aligned}
b_i \backslash B_J &= (a_i, g_i \backslash B_J, d_i \backslash B_J, f_i) \\
g_i \backslash B_J &= g_i \wedge \textstyle\bigwedge_{(a_j, g_j, d_j, f_j) \in B_J, a_i \prec_k a_j} \neg \Diamond_k g_j \\
d_i \backslash B_J &= d_i \wedge g_i \backslash B_J = d_i \wedge \textstyle\bigwedge_{(a_j, g_j, d_j, f_j) \in B_J, a_i \prec_k a_j} \neg \Diamond_k g_j
\end{aligned}
$$

and the $b_j \backslash B_I$'s are defined in a similar manner.

Notice that $\widehat{+}$ preserves strong congruence in the sense that if $s_1 = s_1'$ then $s_1 \widehat{+} s_2 = s_1' \widehat{+} s_2$. This definition introduces $\widehat{+}$ as a macro-notation : any term with priority choice can be expanded into a term with non-deterministic choice (its meaning). The equality of terms with priority choice operators is the strong congruence of their meanings.

From the above definition, it is clear that priority restrictions are applied mutually with respect to actions that are not on the same side of the operator $\widehat{+}$.

Notice that if $a_1 \prec_k a_2$ then in $b_1.s_1 \widehat{+} b_s.s_2 = b_1 \backslash \{b_2\}.s_1 + b_2 \backslash \{b_1\}.s_2 = b_1 \backslash \{b_2\}.s_1 + b_2.s_2$, $a_1$ is disabled if $a_2$ will be enabled within $k$ time units.

Consider the guards $g_1$, $g_2$ of the actions $a_1, a_2$. Figure 3 gives the guard $g'_1 = g_1 \backslash \{b_2\}$ obtained when $g_1$ is restricted by considering the priority orders $a_1 \prec_0 a_2$, $a_1 \prec_1 a_2$, $a_1 \prec_\infty a_2$.

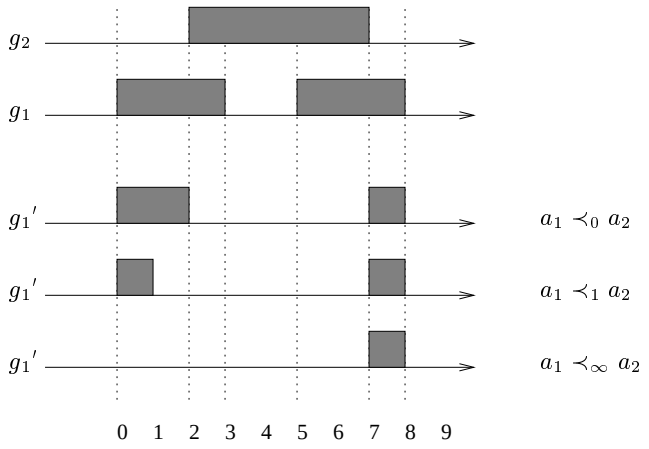



**Fig. 3.** The restricted guard $g_1'$ for different degrees of priority

**Lemma 11.** *For a timed action $b$ and sets of timed actions $B$, $B_1$, $B_2$,*

$$b\backslash\{b\} \cup B = b\backslash B$$
$$(b\backslash B_1)\backslash B_2 = b\backslash(B_1 \cup B_2)$$

*Proof.* Let $b = (a, g, d, f)$.
The first property results from the fact that priority orders are strict.
$b\backslash\{b\} \cup B = (a, g\backslash\{b\} \cup B, d\backslash\{b\} \cup B, f)$
with

$$\begin{aligned} g\backslash\{b\} \cup B &= g \wedge \textstyle\bigwedge_{(a_i,g_i,d_i,f_i)\in\{b\}\cup B,\ a\prec_k a_i} \neg\Diamond_k g_i \\ &= g \wedge \textstyle\bigwedge_{(a_i,g_i,d_i,f_i)\in B,\ a\prec_k a_i} \neg\Diamond_k g_i \\ &= g\backslash B \end{aligned}$$

and

$$d\backslash\{b\} \cup B = d \wedge g\backslash\{b\} \cup B = d \wedge g\backslash B = d\backslash B$$

That is, $b\backslash\{b\} \cup B = b\backslash B$.

For the second property , we have by direct application of definition 5 :

$$(b\backslash B_1)\backslash B_2 = (a, g\backslash B_1, d\backslash B_1, f)\backslash B_2 = (a, (g\backslash B_1)\backslash B_2, (d\backslash B_1)\backslash B_2, f)$$

Let us compute $(g\backslash B_1)\backslash B_2$ :

$$\begin{aligned} (g\backslash B_1)\backslash B_2 &= (g \wedge \textstyle\bigwedge_{(a_i,g_i,d_i,f_i)\in B_1,\ a\prec_k a_i} \neg\Diamond_k g_i)\backslash B_2 \\ &= (g \wedge \textstyle\bigwedge_{(a_i,g_i,d_i,f_i)\in B_1,\ a\prec_k a_i} \neg\Diamond_k g_i) \\ &\quad \wedge \textstyle\bigwedge_{(a_i,g_i,d_i,f_i)\in B_2,\ a\prec_k a_i} \neg\Diamond_k g_i \\ &= g \wedge \textstyle\bigwedge_{(a_i,g_i,d_i,f_i)\in B_1\cup B_2,\ a\prec_k a_i} \neg\Diamond_k g_i \\ &= g\backslash(B_1 \cup B_2) \end{aligned}$$

This implies

$$(d\backslash B_1)\backslash B_2 = (d \wedge g\backslash B_1)\backslash B_2 = (d \wedge g\backslash B_1) \wedge (g\backslash B_1)\backslash B_2$$
$$= d \wedge g\backslash(B_1 \cup B_2) = d\backslash(B_1 \cup B_2)$$

□

It will be shown that the operator $\widehat{+}$ is commutative and $Nil$ is the neutral element. Notice that $\widehat{+}$ is not distributive with respect to $+$ :

$$(b_1.s_1 + b_2.s_2)\widehat{+}b_3.s_3 \neq (b_1.s_1\widehat{+}b_3.s_3) + (b_2.s_2\widehat{+}b_3.s_3) \text{ equivalent to}$$
$$b_1\backslash\{b_3\}.s_1 + b_2\backslash\{b_3\}.s_2 + b_3\backslash\{b_1, b_2\}.s_3 \neq$$
$$b_1\backslash\{b_3\}.s_1 + b_2\backslash\{b_3\}.s_2 + b_3\backslash\{b_1\}.s_3 + b_3\backslash\{b_2\}.s_3$$

In fact, if $a_3$ (the label of $b_3$) is the action with the lowest priority then in $(b_1.s_1 + b_2.s_2)\widehat{+}b_3.s_3$, $b_3$ is restricted jointly by both $b_1$ and $b_2$ , while in $(b_1.s_1\widehat{+}b_3.s_3) + (b_2.s_2\widehat{+}b_3.s_3)$, $b_3$ is restricted separately by $b_1$ and $b_2$. In the latter case $a_3$ cannot be executed when both $a_1$ and $a_2$ are enabled while in the former case $a_3$ cannot be executed when either $a_1$ or $a_2$ is enabled.

However , $\widehat{+}$ is associative as it will be shown in proposition 13. Associativity is an important property which is satisfied due to the adequate definition of priority orders. In particular, the transitivity property is crucial for achieving associativity, as it is shown by the following example.

*Example 12.* Consider the timed terms $p = (b_1.s_1\widehat{+}b_2.s_2)\widehat{+}b_3.s_3$ and $q = b_1.s_1\widehat{+}(b_2.s_2\widehat{+}b_3.s_3)$ with $b_i = (a_i, g_i, d_i, f_i)$, $i = 1, 2, 3$. Suppose that $a_1 \prec_{10} a_2$ and $a_2 \prec_{10} a_3$ and that $a_1 \prec_d a_3$ for some $d \in \mathbf{R}_{\geq \mathbf{0}}$.

Then $p$ and $q$ are respectively equivalent to

$$p = (b_1\backslash\{b_2\})\backslash\{b_3\}.s_1 + b_2\backslash\{b_3\}.s_2 + b_3.s_3$$
$$q = b_1\backslash\{b_2\backslash\{b_3\}, b_3\}.s_1 + b_2\backslash\{b_3\}.s_2 + b_3.s_3$$

For $\widehat{+}$ to be associative, the guard $g_1'$ of $(b_1\backslash\{b_2\})\backslash\{b_3\}$, $g_1' = g_1 \wedge \neg\Diamond_{10} g_2 \wedge \neg\Diamond_d g_3$, and the guard $g_1''$ of $b_1\backslash\{b_2\backslash\{b_3\}, b_3\}$, $g_1'' = g_1 \wedge \neg\Diamond_{10}(g_2 \wedge \neg\Diamond_{10} g_3) \wedge \neg\Diamond_d g_3$ must be equivalent.

Clearly $g_1' \Rightarrow g_1''$. Suppose that $g_1''$ is true at some valuation $v$ and that $d < 20$. In that case, it is possible that $\neg\Diamond_{10}(g_2 \wedge \neg\Diamond_{10} g_3)(v)$ while $\Diamond_{10} g_2(v)$, as it is shown in figure 4. On the contrary, if $d \geq 20$ (the transitivity axiom is satisfied) then $\neg\Diamond_d g_3$ implies that $\neg\Diamond_{10}(g_2 \wedge \neg\Diamond_{10} g_3)$ is equivalent to $\neg\Diamond_{10} g_2$.

**Proposition 13.** *The binary priority operator is associative i.e., for timed actions* $b_i = (a_i, g_i, d_i, f_i)$,

$$((\textstyle\sum_{i\in I} b_i.s_i)\widehat{+}(\sum_{j\in J} b_j.s_j))\ \widehat{+}(\sum_{k\in K} b_k.s_k) =$$
$$(\textstyle\sum_{i\in I} b_i.s_i)\widehat{+}((\sum_{j\in J} b_j.s_j)\widehat{+}(\sum_{k\in K} b_k.s_k)$$

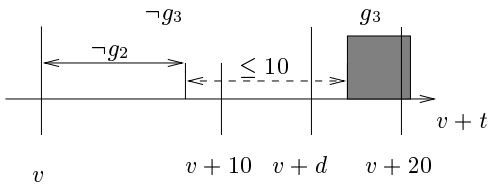



**Fig. 4.** Case $d < 20$

*Proof.* We denote by $B_I$, $B_J$ and $B_K$ respectively the three sets $\{b_i\}_{i\in I}$, $\{b_j\}_{j\in J}$ and $\{b_k\}_{k\in K}$. We have to show the three following equalities :

$$\begin{array}{l}\forall i \in I.\ (b_i \backslash B_J)\backslash B_K = b_i\backslash(\{b_j\backslash B_K\}_{j\in J} \cup \{b_k\backslash B_J\}_{k\in K}) \\ \forall j \in J.\ (b_j \backslash B_I)\backslash B_K = (b_j\backslash B_K)\backslash B_I \\ \forall k \in K.\ b_k\backslash(\{b_i\backslash B_J\}_{i\in I} \cup \{b_j\backslash B_I\}_{j\in J}) = (b_k\backslash B_J)\backslash B_I\end{array}$$

Due to the lemma this is equivalent to

$$\begin{array}{l}\forall i \in I.\ b_i\backslash(B_J \cup B_K) = b_i\backslash(\{b_j\backslash B_K\}_{j\in J} \cup \{b_k\backslash B_J\}_{k\in K}) \\ \forall k \in K.\ b_k\backslash(B_J \cup B_I) = b_k\backslash(\{b_j\backslash B_I\}_{j\in J} \cup \{b_i\backslash B_J\}_{i\in I})\end{array}$$

It is then sufficient to show that :

$$\forall i \in I.\ g_i\backslash(B_J \cup B_K) = g_i\backslash(\{b_j\backslash B_K\}_{j\in J} \cup \{b_k\backslash B_J\}_{k\in K})$$

By definition of $g\backslash B$, this equality can be reduced to

$$\begin{array}{l}\bigwedge_{j\in J,\ a_i \prec_{l_{ij}} a_j} \neg\Diamond_{l_{ij}}(g_j\backslash B_K) \wedge \bigwedge_{k\in K,\ a_i\prec_{l_{ik}} a_k} \neg\Diamond_{l_{ik}}(g_k\backslash B_J) \\ = \bigwedge_{j\in J,\ a_i \prec_{l_{ij}} a_j} \neg\Diamond_{l_{ij}} g_j \wedge \bigwedge_{k\in K,\ a_i\prec_{l_{ik}} a_k} \neg\Diamond_{l_{ik}} g_k\end{array}$$

for every $i$ in $I$.

For a given $i$, we will now prove this by induction on the cardinality of $J\cup K$.

- The case $\mathrm{card}(J\cup K) = 1$ is trivial and left to the reader.
- Let us suppose that the property holds for all $J'$ and $K'$ such that $\mathrm{card}(J'\cup K') = n$,

$$\begin{array}{l}\bigwedge_{j\in J',\ a_i \prec_{l_{ij}} a_j} \neg\Diamond_{l_{ij}}(g_j\backslash B_{K'}) \wedge \bigwedge_{k\in K',\ a_i\prec_{l_{ik}} a_k} \neg\Diamond_{l_{ik}}(g_k\backslash B_{J'}) \\ = \bigwedge_{j\in J',\ a_i \prec_{l_{ij}} a_j} \neg\Diamond_{l_{ij}} g_j \wedge \bigwedge_{k\in K',\ a_i\prec_{l_{ik}} a_k} \neg\Diamond_{l_{ik}} g_k.\end{array}$$

  We will now show that this holds for all $J$ and $K$ such that $\mathrm{card}(J\cup K) = n+1$.
  Let $a$ be an action of least priority in $J\cup K$ : $\forall j \in J\cup K,\ \neg(a_j \prec\!\!\prec a)$
  If $a$ has no priority over $a_i$, then the property to prove is identical to the

assumption. Let us suppose that $a$ has priority over $a_i$, and (without loss of generality) that it appears in $J : a = a_{j_0}$. The property to be shown is then

$$\begin{aligned}
&(\neg\Diamond_{l_{ij_0}} g_{j_0} \backslash B_K) \wedge \textstyle\bigwedge_{j\in(J\backslash\{j_0\}),\ a_i \prec_{l_{ij}} a_j} \neg\Diamond_{l_{ij}} (g_j \backslash B_K) \\
&\qquad\qquad \wedge \textstyle\bigwedge_{k\in K,\ a_i \prec_{l_{ik}} a_k} \neg\Diamond_{l_{ik}} (g_k \backslash B_J) \\
&= \neg\Diamond_{l_{ij_0}} g_{j_0} \wedge \textstyle\bigwedge_{j\in(J\backslash\{j_0\}),\ a_i \prec_{l_{ij}} a_j} \neg\Diamond_{l_{ij}} g_j \wedge \textstyle\bigwedge_{k\in K,\ a_i \prec_{l_{ik}} a_k} \neg\Diamond_{l_{ik}} g_k.
\end{aligned}$$

Since $a_{j_0}$ has the least priority in $J \cup K$, we know that :

$$\forall k \in K.\ g_k \backslash B_J = g_k \backslash (B_J \backslash \{b_{j_0}\})$$

We can use the induction hypothesis on $(J\backslash\{j_0\}) \cup K$ :

$$\begin{aligned}
&(\neg\Diamond_{l_{ij_0}} g_{j_0} \backslash B_K) \wedge \textstyle\bigwedge_{j\in(J\backslash\{j_0\}),\ a_i \prec_{l_{ij}} a_j} \neg\Diamond_{l_{ij}} (g_j \backslash B_K) \\
&\qquad\qquad \wedge \textstyle\bigwedge_{k\in K,\ a_i \prec_{l_{ik}} a_k} \neg\Diamond_{l_{ik}} (g_k \backslash (B_J \backslash \{b_{j_0}\})) \\
&= (\neg\Diamond_{l_{ij_0}} g_{j_0} \backslash B_K) \wedge \textstyle\bigwedge_{j\in(J\backslash\{j_0\}),\ a_i \prec_{l_{ij}} a_j} \neg\Diamond_{l_{ij}} g_j \wedge \textstyle\bigwedge_{k\in K,\ a_i \prec_{l_{ik}} a_k} \neg\Diamond_{l_{ik}} g_k.
\end{aligned}$$

Since $\Diamond_k$ is distributive with respect to disjunction and since $\Diamond_l \Diamond_k g = \Diamond_{l+k} g$, we have :

$$\begin{aligned}
\textstyle\bigwedge_{k\in K,\ a_i \prec_{l_{ik}} a_k} \neg\Diamond_{l_{ik}} g_k &= \textstyle\bigwedge_{k\in K,\ a_i \prec_{l_{ik}} a_k} \neg\Diamond_{l_{ij_0}} \Diamond_{l_{j_0 k}} g_k \\
&= \neg\Diamond_{l_{ij_0}} \textstyle\bigvee_{k\in K,\ a_i \prec_{l_{ik}} a_k} \Diamond_{l_{j_0 k}} g_k \\
&= \neg\Diamond_{l_{ij_0}} \textstyle\bigvee_{k\in K,\ a_{j_0} \prec_{l_{ij_0}} a_k} \Diamond_{l_{j_0 k}} g_k
\end{aligned}$$

Let us take $G = \bigvee_{k\in K,\ a_{j_0} \prec_{l_{ij_0}} a_k} \Diamond_{l_{j_0 k}} g_k$. Then, the following holds :

$$\begin{aligned}
&(\neg\Diamond_{l_{ij_0}} g_{j_0} \backslash B_K) \wedge \textstyle\bigwedge_{j\in(J\backslash\{j_0\}),\ a_i \prec_{l_{ij}} a_j} \neg\Diamond_{l_{ij}} (g_j \backslash B_K) \\
&\qquad\qquad \wedge \textstyle\bigwedge_{k\in K,\ a_i \prec_{l_{ik}} a_k} \neg\Diamond_{l_{ik}} (g_k \backslash (B_J \backslash \{b_{j_0}\})) \\
&= (\neg\Diamond_{l_{ij_0}} (g_{j_0} \wedge \neg G)) \wedge \textstyle\bigwedge_{j\in(J\backslash\{j_0\}),\ a_i \prec_{l_{ij}} a_j} \neg\Diamond_{l_{ij}} g_j \wedge \neg\Diamond_{l_{ij_0}} G \\
&= (\neg\Diamond_{l_{ij_0}} ((g_{j_0} \wedge \neg G) \vee G)) \wedge \textstyle\bigwedge_{j\in(J\backslash\{j_0\}),\ a_i \prec_{l_{ij}} a_j} \neg\Diamond_{l_{ij}} g_j \\
&= (\neg\Diamond_{l_{ij_0}} (g_{j_0} \vee G)) \wedge \textstyle\bigwedge_{j\in(J\backslash\{j_0\}),\ a_i \prec_{l_{ij}} a_j} \neg\Diamond_{l_{ij}} g_j \\
&= (\neg\Diamond_{l_{ij_0}} g_{j_0}) \wedge \neg\Diamond_{l_{ij_0}} G \wedge \textstyle\bigwedge_{j\in(J\backslash\{j_0\}),\ a_i \prec_{l_{ij}} a_j} \neg\Diamond_{l_{ij}} g_j \\
&= \neg\Diamond_{l_{ij_0}} g_{j_0} \wedge \textstyle\bigwedge_{j\in(J\backslash\{j_0\}),\ a_i \prec_{l_{ij}} a_j} \neg\Diamond_{l_{ij}} g_j \wedge \textstyle\bigwedge_{k\in K,\ a_i \prec_{l_{ik}} a_k} \neg\Diamond_{l_{ik}} g_k
\end{aligned}$$

QED.

□

The above proposition allows the definition of a n-ary priority choice operator. We denote by $\widehat{\sum}_{i\in I} b_i.s_i$ the term obtained by combining the terms $\{b_i.s_i\}_{i\in I}$ by means of $\widehat{+}$.

**Definition 14.** Let $\widehat{P}(B)$ be the set of the well-guarded terms built from $Nil$ and a set of variables VAR by using prefixing by timed actions of $B$, priority choice and recursion.

**Proposition 15.** *On $\widehat{P}(B)$, the priority choice operator $\widehat{+}$ is commutative, associative, idempotent and Nil is the neutral element.*

*Proof.* Directly from the definition, $\widehat{+}$ is commutative and $Nil$ is the neutral element. It is also associative from the previous proposition. It is trivial that $p\widehat{+}p = p$ for all term of the form $p = b.s$, for some timed action $b$ and some term $s$. By associativity of $\widehat{+}$, this equality can be generalized to all terms $p$, that is, $\widehat{+}$ is idempotent. □

This result allows to consider $\widehat{+}$ not only as a macro-notation but also as a basic operator.

The following two propositions deal with the correspondence between $\widehat{P}(B)$ and $P(B)$ ant its properties.

**Proposition 16.** Reduction to non-deterministic choice
*For any finite set of terms $\{b_i.s_i\}_{i\in I}$ with $b_i = (a_i, g_i, d_i, f_i)$*

$$\widehat{\sum_{i\in I} b_i.s_i} = \sum_{i\in I} b'_i.s_i$$

*with $b'_i = (a_i, g'_i, d'_i, f_i)$, $g'_i = g_i \wedge \bigwedge_{a_i \prec_k a_j} \neg\Diamond_k g_j$ and $d'_i = d_i \wedge g'_i$. That is $b'_i = b_i \backslash \{b_j\}_{j\in I}$.*

*Proof.* The result is immediate by induction on $I$, with the help of the two previous propositions. But we need also to verify that time reactivity is preserved when priority choice operators are applied to systems with left-closed deadlines. We only have to check that when by restriction of some guard a left-open deadline is obtained then the rising edge of the deadline is implied by a guard of some action of higher priority. This also is immediate by induction on $I$ since, by definition, a deadline can only be restricted to the left if it intersects the guard of an action of higher priority. □

**Proposition 17.** Activity preservation
*If $\widehat{\sum}_{i\in I} b_i.s_i = \sum_{i\in I} b'_i.s_i$ as in proposition 16, then the following properties hold between the guards $g_i$ of $b_i$ and the restricted guards $g'_i$ of $b'_i$.*

1. $\Diamond g_i \Rightarrow \Diamond(g'_i \vee \bigvee_{a_i \prec\prec a_j} g'_j)$, *for any* $i \in I$
2. $\Diamond \bigvee_{i\in I} g_i = \Diamond \bigvee_{i\in I} g'_i$

*Proof.* The proof of the second property is a direct application of associativity of $\widehat{+}$. Let us consider a timed action $b = (a, g, d, f)$ with infinitely less priority than all actions in $I$ ($\forall i \in I.\ a \prec_\infty a_i$) and a maximal guard ($g = true$). We have

$$b.s\widehat{+}\widehat{\sum_{i\in I} b_i.s_i} = b.s\widehat{+}\sum_{i\in I} b'_i.s_i$$

by application of proposition 16. The restricted guard $g'$ of $b$ is $g' = true\backslash\{b'_i\}_{i\in I} = true\backslash\{b_i\}_{i\in I}$ which can be written $g' = \bigwedge_{i\in I} \neg\Diamond g'_i = \bigwedge_{i\in I} \neg\Diamond g_i$ and gives the result.

The first property is obtained by considering in the previous example instead of $\widehat{\sum}_{i\in I} b_i.s_i$ the term $b_i.s_i \widehat{+} \widehat{\sum}_{j\in I, a_i \prec\prec a_j} b_j.s_j$ for a given $i \in I$.

$$b.s\widehat{+}(b_i.s_i\widehat{+}\widehat{\sum_{j\in I, a_i\prec\prec a_j} b_j.s_j}) = b.s\widehat{+}(b'_i.s_i + \sum_{j\in I, a_i\prec\prec a_j} b'_j.s_j)$$

The guards of $b$ in the two terms are equal. We obtain $\Diamond(g_i \vee \bigvee_{a_i\prec\prec a_j} g_j) = \Diamond({g'}_i \vee \bigvee_{a_i\prec\prec a_j} g'_j)$. Notice that the restricted guards $g'_i$ and $g'_j$, for $j \in I, a_i \prec\prec a_j$ are the same as the guards of $b'_i$ and $b'_j$ for $j \in I, a_i \prec\prec a_j$ in $\sum_{i\in I} b_i.s_i$. The property follows immediately. □

The first property means that if action $a_i$ can occur in the non-prioritized choice then either $a_i$ can occur in the prioritized choice or some action of higher priority.
The second property simply says that $\widehat{\sum}$ preserves activity of components : if some action can be executed in the non-prioritized choice then some action can be executed in the prioritized choice and vice versa.

## 4 Parallel Composition

The results of this section show that non-deterministic choice is a special case of priority choice when the priority order is empty. Priority choice is actually a generalization of non-deterministic choice and for this reason we consider it as the choice operator, in the sequel. This allows to describe behaviors depending on a priority order. More precisely, given a priority order $\prec$ on a set of actions $A$, and the corresponding set of timed actions $B$, we will only consider terms of the associated language $\widehat{P}(B)$.

In this section, we propose a general method for the definition of parallel composition operators for timed systems as an extension of parallel composition for untimed systems.

### 4.1 Parallel composition of untimed systems

We consider that for parallel composition of untimed terms the following framework is given.

- The action vocabulary $A$ is provided with an operator $\mid$ such that $(A, \mid)$ is a commutative semi-group with a distinguished absorbing element $\perp \in A$. Words of this monoid represent the action resulting from the synchronization of their elements. The absorbing element $\perp$ means impossibility of synchronization.
- A *parallel composition operator* $\|$ on terms which is supposed to be associative, commutative, has $Nil$ as neutral element and is defined by an expansion

rule of the form:

If $p_1 = \sum_{i \in J} \ a_i.s_i$ and $p_2 = \sum_{j \in J} \ a_j.s_j$ then

$$p_1 \| p_2 = \sum_{i \in I'} \ a_i.(s_i \| p_2) \ + \sum_{j \in J'} \ a_j.(s_j \| p_1) \ + \sum_{(i,j) \in I \times J} \ a_i | a_j.(s_i \| s_j) \quad (\alpha)$$

where $I'$ and $J'$ are subsets of $I$ and $J$ respectively.

The first two summands correspond to behaviors starting with interleaving actions. The sets of interleaving actions may be empty, depending on the semantics of $\|$. The third summand contains terms with synchronization transitions where only terms such that $a_i | a_j \neq \perp$ appear.

When such a parallel composition operator is used to compose sequential systems, it is important to combine interleaving and synchronization so as to satisfy two often conflicting requirements:

- *activity preservation*, that is, if in one of the components some action is enabled, then in the product some action is enabled too.
- *maximal progress*, that is, when in the product both synchronization and interleaving transitions are enabled, synchronization is taken.

Clearly, it is easy to satisfy each requirement separately.

- If all the actions interleave ($I = I', J = J'$ in the expansion rule) then activity is preserved. However, in this case to achieve maximal progress the description language should provide with mechanisms for eliminating dynamically all the interleaving transitions that are systematically introduced. This is the approach adopted in languages such as CCS [Mil89] where all the actions interleave and a global restriction operator is often applied to prune off interleaving transitions.
- Maximal progress can be easily achieved by not allowing interleaving of actions that must synchronize. However, in this case there is an obvious risk of deadlock when the synchronization actions do not match. This point of view is adopted in languages such as CSP [Hoa85], where actions are partitioned into two classes, synchronizing and interleaving actions.

We show that for timed systems a parallel composition operation can be defined preserving process activity and maximal progress due to the possibility of controlling waiting times by means of priority choice operators.

### 4.2 Parallel composition of timed systems

We extend the parallel composition operator $\|$ to timed systems in the following manner:

**extension of** $|$ We assume that the operator $|$ can be extended component-wise on the set $B$ of timed actions $b$ of the form $(a, g, d, f)$ where $a \in A$, in such a manner that $(B, |)$ is a commutative semi-group with a distinguished absorbing element $\perp$. We take $(\perp, g, d, f) = \perp$ for any $g$, $d$, and $f$.
As ambiguity is resolved by the context, for sake of simplicity, we overload the notation for $|$ and $\perp$.

**extension of the priority order** If $\prec$ is a priority order on $A$ we suppose that it is preserved by $|$

$$\forall a_1, a_2, a_3 \in A \,.\, a_1 \prec_k a_2 \;\text{ implies }\; a_1|a_3 \prec_k a_2|a_3$$

**extension of** $\|$ The parallel composition operator $\|$ for timed systems is defined by extending the expansion rule $(\alpha)$ to timed terms, where $b_i$ is the timed action associated with $a_i$.

$$\text{If } p_1 = \widehat{\sum_{i\in I}} b_i.s_i \;\text{ and }\; p_2 = \widehat{\sum_{j\in J}} b_j.s_j \;\text{ then}$$

$$p_1 \,\|\, p_2 = \widehat{\sum_{i\in I'}} b_i.(s_i \,\|\, p_2) \;\widehat{+}\; \widehat{\sum_{j\in J'}} b_j.(p_1 \,\|\, s_j) \;\widehat{+}\; \widehat{\sum_{(i,j)\in I\times J}} b_i|b_j.(s_i \,\|\, s_j)$$

**Proposition 18.** *The parallel composition operator $\|$ defined above is associative, commutative, distributive with respect to $\widehat{+}$ and has $Nil$ as neutral element.*

*Proof.* The proof is standard and similar to the one given in [Mil83]. It is based on the uniqueness of solution of well-guarded equations and on properties of $\widehat{+}$. □

**Proposition 19.** *If all the actions interleave then $\|$ preserves activity. That is, if $g_i$ are the guards of $b_i$, $i \in I \cup J$, in the expansion rule, $g_i'$ are the restricted guards of the interleaving actions, $i \in I \cup J$ and $g_{ij}$ are the guards of $b_i|b_j$, $(i,j) \in I \times J$, then*

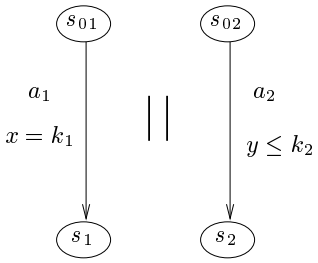
$$\Diamond g_i \Rightarrow \Diamond(g_i' \vee \textstyle\bigvee_{j\in J} g_{ij})$$
$$\Diamond(\textstyle\bigvee_{i\in I} g_i \vee \bigvee_{j\in J} g_j) = \Diamond(\bigvee_{i\in I} g_i' \vee \bigvee_{j\in J} g_j' \vee \bigvee_{i,j\in I\times J} g_{ij})$$

*Proof.* If in the expansion rule priority choice is replaced by non-deterministic choice, activity is trivially preserved due to the presence of interleaving actions. Proposition 17 says that replacing non-deterministic choice by priority choice preserves activity. □

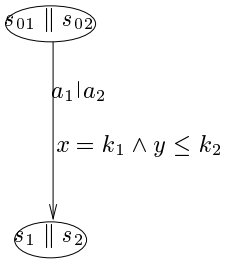
This proposition guarantees activity preservation. If some action is possible in a component, then in the product, either this action can interleave or it can participate to a synchronization.

To achieve maximal progress in the expansion rule, it is sufficient to consider the priority order which gives infinite priority to synchronizations :

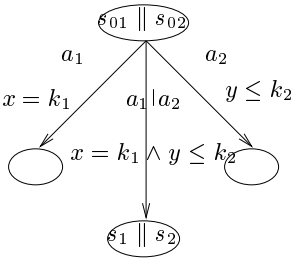
$$\forall a_1, a_2 \in A \,.\, a_1|a_2 \neq \perp \;\text{ implies }\; a_1 \prec_\infty a_1|a_2 \;\text{ and }\; a_2 \prec_\infty a_1|a_2$$



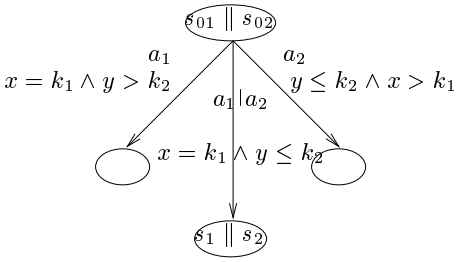

a. The components




b. No interleaving




c. Interleaving without priorities




d. Interleaving and priorities

**Fig. 5.** Parallel composition

*Example 20.* Consider $b_1.s_1 \parallel b_2.s_2$ with $b_i = (a_i, g_i, d_i, id)$ such that $a_1|a_2 \neq \bot$, $g_1 = (x = k_1)$, $g_2 = (y \leq k_2)$ and the synchronization guard is $g_1|g_2 = g_1 \wedge g_2$.

If $a_1$ and $a_2$ do not interleave, then $b_1.s_1 \parallel b_2.s_2 = (b_1|b_2)(s_1 \parallel s_2)$. We have maximal progress but if we start from states such that $\neg\Diamond((x = k_1) \wedge (y \leq k_2))$, we have a deadlock (figure 5b).

If actions $a_1$ and $a_2$ interleave and there is no priority between $a_1|a_2$ and these actions, then activity is preserved but either of the interleaving actions can be taken when synchronization is possible (figure 5c).

Finally, if actions $a_1$ and $a_2$ interleave and $a_1 \prec_\infty a_1|a_2$, $a_2 \prec_\infty a_1|a_2$ then activity is preserved due to proposition 19. Furthermore, we have maximal progress because the guards of the interleaving actions are respectively $g_1 \wedge \neg\Diamond(g_1 \wedge g_2)$ and $g_2 \wedge \neg\Diamond(g_1 \wedge g_2)$, which means that they can be taken only if the synchronization is disabled forever.

## 5 The Algebraic Framework

In this section we develop an algebraic framework for the specification of timed systems which takes into account the structure of timed actions. We study a simple algebra for the composition of timed actions and deduce two classes of laws for terms. The first class contains laws modulo strong congruence, resulting from the properties of priority choice and the definition of parallel composition operators. The second class contains laws reflecting properties of timed actions and preserving timed bisimulation,

### 5.1 Composition of Guards and Deadlines

We show how the commutative semi-group $(B, |)$ can be defined.

Assume that the composition of timed actions $b_i = (a_i, g_i, d_i, f_i), i = 1, 2$, is a timed action of the form $b_1|b_2 = (a_1|a_2, g_1|g_2, d_1|d_2, f_1|f_2)$. For sake of simplicity we use the same notation, $|$ , to denote the composition of timed actions, actions, guards, deadlines and jumps.

- For the composition of the guards, we suppose that the guard $g_1|g_2$ is defined as a monotonic function of $g_1$ and $g_2$ called *synchronization mode*, of the general form
$$g_1|g_2 = (g_1 \wedge m(g_2)) \vee (m(g_1) \wedge g_2)$$
where $m$ is a function such that:
  - $\forall g \;.\; g \Rightarrow m(g)$
  - $\forall g, g' \;.\; m(g \vee g') = m(g) \vee m(g')$
  - $\forall g, g' \;.\; m(g|g') = m(g) \wedge m(g')$

- For a given synchronization guard $g_1|g_2$, the associated deadline $d_1|d_2$ must be such that $d_1|d_2 \Rightarrow g_1|g_2$, to preserve time reactivity. On the other hand, it is desirable to preserve urgency which means $d_1|d_2 \Rightarrow d_1 \vee d_2$. For maximal urgency and time reactivity we take $d_1|d_2 = (g_1|g_2) \wedge (d_1 \vee d_2)$. Notice that

this is sufficient to ensure time reactivity when $g_1|g_2$ is left-closed since the deadline $d_1|d_2$ is then left-closed. This is the case for the four synchronization modes considered below.
- The definition of $f_1|f_2$ does not pose particular problems. An associative and commutative operator $|$ can be defined on jumps (consider for instance, the easy case where synchronizing actions transform disjoint state spaces).

**Proposition 21.** *For guards $g_1$, $g_2$ and $|$ synchronization mode,*

$$g_1|g_2 = g_2|g_1$$
$$(g_1|g_2)|g_3 = g_1|(g_2|g_3)$$
$$(g_1 \vee g_2)|g_3 = (g_1|g_3) \vee (g_2|g_3)$$
$$g_1 \wedge g_2 \Rightarrow g_1|g_2 \Rightarrow g_1 \vee g_2$$

*Proof.* Commutativity of $|$ follows directly from its definition.
Associativity is a simple application of the definition and the properties of $m$ :

$$\begin{aligned}(g_1|g_2)|g_3 &= m(g_1|g_2) \wedge g_3 \vee (g_1|g_2) \wedge m(g_3) \\ &= m(g_1) \wedge m(g_2) \wedge g_3 \vee m(g_1) \wedge g_2 \wedge m(g_3) \vee g_1 \wedge m(g_2) \wedge m(g_3)\end{aligned}$$

Due to commutativity of $|$, this is equal to $g_1|(g_2|g_3)$.
Distributivity with respect to disjunction is obtained by :

$$\begin{aligned}(g_1 \vee g_2)|g_3 &= m(g_1 \vee g_2) \wedge g_3 \vee (g_1 \vee g_2) \wedge m(g_3) \\ &= (m(g_1) \vee m(g_2)) \wedge g_3 \vee g_1 \wedge m(g_3) \vee g_2 \wedge m(g_3) \\ &= (m(g_1) \wedge g_3 \vee g_1 \wedge m(g_3)) \vee (m(g_2) \wedge g_3 \vee g_2 \wedge m(g_3)) \\ &= (g_1|g_3) \vee (g_2|g_3)\end{aligned}$$

The last property is derived from $g_1 \Rightarrow m(g_1)$ and $g_2 \Rightarrow m(g_2)$, knowing that $g_1|g_2 = m(g_1) \wedge g_2 \vee g_1 \wedge m(g_2)$ :

$$g_1 \wedge g_2 \vee g_1 \wedge g_2 \Rightarrow g_1|g_2$$

Moreover $g_1 \wedge m(g_2) \Rightarrow g_1$ and $g_2 \wedge m(g_1) \Rightarrow g_2$ imply :

$$g_1|g_2 \Rightarrow g_1 \vee g_2$$

□

The above properties imply that synchronization may occur only if at least one of the synchronizing actions is enabled. Furthermore, if both synchronizing actions are enabled at a state then synchronization is enabled. Distributivity of the composition of guards with respect to disjunction is an important property as parallel composition distributes over choice operator. More precisely, if $TS'$ is the timed system $TS$ where we replace a transition $s \xrightarrow{(a,g,d,f)} s'$ by the two transitions $s \xrightarrow{(a,g_1,d_1,f)} s'$ and $s \xrightarrow{(a,g_2,d_2,f)} s'$ such that $g = g_1 \vee g_2$ et $d = d_1 \vee d_2$ we would like that the parallel composition of $TS$ and $TS'$ with another timed system yields timed bisimilar systems.

In previous papers [BST97] we use the following synchronization modes for their practical interest:

- **and-synchronization** where $g_1|g_2 = g_1 and\ g_2 = g_1 \wedge g_2$.
- **max-synchronization** where $g_1|g_2 = g_1 max\ g_2 = (\diamondsuit\, g_1 \wedge g_2) \vee (g_1 \wedge \diamondsuit\, g_2)$. This condition characterizes synchronization with waiting.
- **min-synchronization** where $g_1|g_2 = g_1 min\ g_2 = (\diamondsuit g_1 \wedge g_2) \vee (g_1 \wedge \diamondsuit g_2)$. This condition characterizes synchronization by anticipation, in the sense that synchronization occurs when one of the two actions is enabled provided that the other will be enabled in the future.
- **or-synchronization** where $g_1|g_2 = g_1 or\ g_2 = g_1 \vee g_2$

It is trivial to check that the above functions are indeed synchronization modes.

### 5.2 Laws for Extended Guards

We call *extended guard* any pair of predicates $G = (g, d)$ such that $d \Rightarrow g$. We extend the equivalence on predicates to equivalence on extended guards : $(g_1, d_1) = (g_2, d_2)$ if $g_1 = g_2$ and $d_1 = d_2$.

If $G_i = (g_i, d_i)$, for $i = 1, 2$, are two extended guards and $|$ is a synchronization mode, we take $G_1|G_2 = (g_1|g_2, g_1|g_2 \wedge (d_1 \vee d_2))$.

**Proposition 22.** *If* $g_1|g_2 = (g_1 \wedge m(g_2)) \vee (m(g_1) \wedge g_2)$ *and* $G_i = (g_i, d_i)$, *for* $i = 1, 2$, *then* $G_1|G_2 = (g_1|g_2, (d_1 \wedge m(g_2)) \vee (m(g_1) \wedge d_2))$.

*Proof.* By definition, $G_1|G_2 = (g_1|g_2, (g_1|g_2) \wedge (d_1 \vee d_2))$. Compute the deadline :

$$(g_1|g_2) \wedge (d_1 \vee d_2) = (m(g_1) \wedge g_2 \vee g_1 \wedge m(g_2)) \wedge (d_1 \vee d_2)$$

Since $d \Rightarrow g \Rightarrow m(g)$ for any extended guard $(g, d)$, this can be reduced to :

$$\begin{aligned}(g_1|g_2) \wedge (d_1 \vee d_2) &= d_1 \wedge g_2 \vee d_1 \wedge m(g_2) \vee m(g_1) \wedge d_2 \vee g_1 \wedge d_2 \\ &= d_1 \wedge m(g_2) \vee m(g_1) \wedge d_2\end{aligned}$$

□

This proposition says that the deadline of the synchronization guard has the same form as the synchronization guard. The following are useful laws that follow as a direct application of the proposition for $G_i = (g_i, d_i)$, $i = 1, 2$.

$G_1 and\ G_2 = (g_1 \wedge g_2, d_1 \wedge g_2 \vee g_1 \wedge d_2)$
$G_1 or\ G_2 = (g_1 \vee g_2, d_1 \vee d_2)$
$G_1 max\ G_2 = (g_1 max\ g_2, (d_1 \wedge \diamondsuit\, g_2) \vee (\diamondsuit\, g_1 \wedge g_2))$
$G_1 min\ G_2 = (g_1 min\ g_2, (d_1 \wedge \diamondsuit g_2) \vee (\diamondsuit g_1 \wedge g_2))$

**Proposition 23.** *For extended guards* $G_i = (g_i, d_i)$, $i = 1, 2, 3$, *and* $|$ *a synchronization mode, the following laws hold*

$(G_1|G_2) = (G_2|G_1)$
$(G_1|G_2)|G_3 = G_1|(G_2|G_3)$
$(G_1 or\ G_2)|G_3 = (G_1|G_3) or\ (G_2|G_3)$

*Proof.* From the previous proposition, Commutativity trivially follows.
Let us prove associativity. Remember that by definition of $m$, $m(g_1|g_2) = m(g_1) \wedge m(g_2)$. We have :

$$\begin{aligned}(G_1|G_2)|G_3 &= ((g_1|g_2), m(g_1) \wedge d_2 \vee d_1 \wedge m(g_2))|(g_3, d_3) \\ &= ((g_1|g_2)|g_3, m(g_1|g_2) \wedge d_3 \vee \\ &\quad (m(g_1) \wedge d_2 \vee d_1 \wedge m(g_2)) \wedge m(g_3)) \\ &= ((g_1|g_2)|g_3, m(g_1) \wedge m(g_2) \wedge d_3 \\ &\quad \vee m(g_1) \wedge d_2 \wedge m(g_3) \vee d_1 \wedge m(g_2) \wedge m(g_3))\end{aligned}$$

As the operator $|$ is associative on guards, this is equal to $G_1|(G_2|G_3)$.
The last equality is derived from the definitions :

$$\begin{aligned}(G_1 \, or \, G_2)|G_3 &= (g_1 \vee g_2, d_1 \vee d_2)|(g_3, d_3) \\ &= ((g_1 \vee g_2)|g_3, m(g_1 \vee g_2) \wedge d_3 \vee (d_1 \vee d_2) \wedge m(g_3) \\ &= ((g_1|g_3) \vee (g_2|g_3), \\ &\quad m(g_1) \wedge d_3 \vee m(g_2) \wedge d_3 \vee d_1 \wedge m(g_3) \vee d_2 \wedge m(g_3) \\ &= (g_1|g_3, m(g_1) \wedge d_3 \vee d_1 \wedge m(g_3)) \, or \\ &\quad (g_2|g_3, m(g_2) \wedge d_3 \vee d_2 \wedge m(g_3)) \\ &= (G_1|G_3) \, or \, (G_2|G_3)\end{aligned}$$

□

Notice that any expression involving extended guards and synchronization modes can be reduced to an equivalent extended guard.

### 5.3 Laws for Timed Actions

We naturally lift the structure of extended guards to timed actions $b = (a, G, f)$. For $b_i = (a_i, G_i, f_i), i = 1, 2$, we take

- $(a_1, G_1, f_1) = (a_2, G_2, f_2)$ if $a_1 = a_2$, $G_1 = G_2$ and $f_1 = f_2$.
- $\perp = (\perp, G, f)$

**Proposition 24.** *Let $B$ be a set of timed actions on a vocabulary $A$ as in paragraph 4.2. $(B, |)$ is a commutative semi-group with absorbing element $\perp$ where $b_1|b_2 = (a_1|a_2, G_1|G_2, f_1|f_2)$, for $b_i = (a_i, G_i, f_i)$, $i = 1, 2$, and $|$ is a given synchronization mode in $G_1|G_2$.*

*Proof.* From the above definitions and proposition 23, it follows that $|$ is associative and commutative on each component of the timed actions. So it is commutative and associative on timed actions. Moreover, the timed action $\perp$ inherits the absorption property of the action $\perp$. □

The above proposition holds for a given synchronization mode. It can be easily extended to allow composition of timed actions with different synchronization modes under the following conditions.

Suppose that a partial function $\mu$ is given from $A$ into the set of modes. If $\mu$ is defined for $a \in A$, $\mu(a)$ denotes the synchronization mode associated with $a$. We require that actions with different synchronization modes cannot synchronize, that is, $\mu(a_1) \neq \mu(a_2)$ implies $a_1 | a_2 = \bot$.

It is easy to check that $(B, |)$ with $b_1 | b_2 = (a_1 | a_2, G_1 \mu(a_1) G_2, f_1 | f_2)$ is a commutative semi-group with $\bot$ as absorbing element. We consider in the sequel, that parallel composition of timed systems is defined in terms of such a general synchronization function.

### 5.4 Laws for Timed Systems

**Proposition 25.** *The congruence induced by the following laws on timed systems on $(B, |)$ is compatible with timed bisimulation, i.e. if two terms are congruent then they are timed bisimilar.*

- $\widehat{+}$ *is associative, commutative, idempotent, and* $Nil$ *is the neutral element.*
- $\|$ *is associative, commutative, distributive with respect to* $\widehat{+}$*, and* $Nil$ *is the neutral element.*
- $\bot.s = Nil$
- $(a, G_1 \, or \, G_2, f).s = (a, G_1, f).s \widehat{+} (a, G_2, f).s$ *(which means that any timed transition is equivalent to two timed transitions with the same label and jump, and such that the disjunction of their guards is equal to its guard)*
- *if all actions interleave and* $b$ *is such that* $b | b_j = \bot$ *for any timed action* $b_j$ *in* $B$ *then*

$$b.s \widehat{+} \widehat{\sum_{i \in I}} b_i.s_i = b \backslash \{b_i\}_{i \in I}.s \widehat{+} \widehat{\sum_{i \in I}} b_i.s_i$$

*Proof.* The proof is carried into two steps. The first step consists in checking that the laws are compatible with timed bisimulation; this is trivial and left to the reader. The second step consists in checking that the induced congruence is compatible with timed bisimulation, that is if $t_1 = t'_1$ and $t_2 = t'_2$, due to one of the laws, then $t_1 \widehat{+} t_2$ and $t_1 \parallel t_2$ are respectively timed bisimilar to $t'_1 \widehat{+} t'_2$ and $t'_1 \parallel t'_2$. Using the fact that we consider equivalences, we will only show that if $t_1 = t_2$, then for any timed system $t$, $t_1 \widehat{+} t$ is timed bisimilar to $t_2 \widehat{+} t$ and $t_1 \parallel t$ is timed bisimilar to $t_2 \parallel t$.

If $t_1 = t_2$ then due to properties of $\widehat{+}$ or $\|$ this property holds (see properties in sections 3 and 4, respectively).

For, the rest of the laws, it is trivial to check that if $t_1 = t_2$ then for any $t$, $t_1 \widehat{+} t$ is timed bisimilar to $t_2 \widehat{+} t$. It is also easy to check that for any $s$ and $t$, $\bot.s \parallel t$ is timed bisimilar to $Nil \parallel t$ (which is equal to $t$), and if $b_1 = b_2$ then $b_1.s \parallel t$ is timed bisimilar to $b_2.s \parallel t$. We will only consider the last two cases.

Knowing that $(a, G_1 \, or \, G_2, f).s = (a, G_1, f).s \widehat{+} (a, G_2, f).s$, consider the term $(a, G_1 \, or \, G_2, f).s \parallel p$ with $p = \widehat{\sum}_{i \in I} b_i.s_i$ and $b_i = (a_i, G_i, f_i)$, $i \in I$. From

properties of parallel composition and choice operator we have

$$\begin{aligned}
(a, G_1 \mathit{or}\ G_2, f).s \parallel p &= (a, G_1 \mathit{or}\ G_2, f).(s \parallel p)\widehat{+}\widehat{\sum}_{i\in I} b_i.((a, G_1 \mathit{or}\ G_2, f).s \parallel s_i)\widehat{+}\\
&\quad \widehat{\sum}_{i\in I}(a, G_1 \mathit{or}\ G_2, f)|b_i.(s \parallel s_i)\\
&= (a, G_1, f).(s \parallel p)\widehat{+}(a, G_2, f).(s \parallel p)\widehat{+}\\
&\quad \widehat{\sum}_{i\in I} b_i.((a, G_1 \mathit{or}\ G_2, f).s \parallel s_i)\widehat{+}\\
&\quad \widehat{\sum}_{i\in I}(a|a_i, (G_1 \mathit{or}\ G_2)|G_i, f|f_i).(s \parallel s_i)\\
&= (a, G_1, f).(s \parallel p)\widehat{+}(a, G_2, f).(s \parallel p)\widehat{+}\\
&\quad \widehat{\sum}_{i\in I} b_i.((a, G_1 \mathit{or}\ G_2, f).s \parallel s_i)\widehat{+}\\
&\quad \widehat{\sum}_{i\in I}(a|a_i, (G_1|G_i) \mathit{or}\ (G_2|G_i), f|f_i).(s \parallel s_i)\\
&= (a, G_1, f).(s \parallel p)\widehat{+}(a, G_2, f).(s \parallel p)\widehat{+}\\
&\quad \widehat{\sum}_{i\in I} b_i.((a, G_1 \mathit{or}\ G_2, f).s \parallel s_i)\widehat{+}\\
&\quad \widehat{\sum}_{i\in I}(a|a_i, (G_1|G_i), f|f_i).(s \parallel s_i)\widehat{+}\\
&\quad \widehat{\sum}_{i\in I}(a|a_i, (G_2|G_i), f|f_i).(s \parallel s_i).
\end{aligned}$$

For $((a, G_1, f).s\widehat{+}(a, G_2, f).s) \parallel p$ we get the same terms with the difference that in the second summand $(a, G_1 \mathit{or}\ G_2, f).s$ is replaced by $(a, G_1, f).s\widehat{+}(a, G_2, f).s$. The rest of the proof is standard and closely follows techniques given [Mil83,Mil89] by using uniqueness of the solution of well-guarded equations modulo strong congruence.

Suppose now that all action interleave and $b$ does not synchronize (for any $b_j \in B$, $b|b_j = \bot$). Consider the term $(b\backslash\{b_i\}_{i\in I}.s\widehat{+}\widehat{\sum}_{i\in I} b_i.s_i) \parallel p$ with $p = \widehat{\sum}_{j\in J} b_j.s_j$. We have :

$$\begin{aligned}
&(b\backslash\{b_i\}_{i\in I}.s\ \widehat{+}\widehat{\sum}_{i\in I} b_i.s_i) \parallel p\\
&\qquad = b\backslash\{b_i\}_{i\in I}.(s \parallel p)\widehat{+}\widehat{\sum}_{i\in I} b_i.(s_i \parallel p)\\
&\qquad\quad \widehat{+}\widehat{\sum}_{j\in J} b_j.((b\backslash\{b_i\}_{i\in I}\widehat{+}\widehat{\sum}_{i\in I} b_i.s_i) \parallel s_j)\\
&\qquad\quad \widehat{+}\widehat{\sum}_{j\in J}(b\backslash\{b_i\}_{i\in I})|b_j.(s \parallel s_j)\widehat{+}\widehat{\sum}_{i,j\in I\times J}(b_i|b_j).(s_i \parallel s_j)\\
&\qquad = b.(s \parallel p)\widehat{+}\widehat{\sum}_{i\in I} b_i.(s_i \parallel p)\widehat{+}\widehat{\sum}_{j\in J} b_j.((b\backslash\{b_i\}_{i\in I}\widehat{+}\widehat{\sum}_{i\in I} b_i.s_i) \parallel s_j)\\
&\qquad\quad \widehat{+}\widehat{\sum}_{j\in J}\bot.(s \parallel s_j)\widehat{+}\widehat{\sum}_{i,j\in I\times J}(b_i|b_j).(s_i \parallel s_j).
\end{aligned}$$

For $(b\widehat{+}\widehat{\sum}_{i\in I} b_i.s_i) \parallel p$ we get the same terms with the difference that in the third summand $b\backslash\{b_i\}_{i\in I}.s\widehat{+}\widehat{\sum}_{i\in I} b_i.s_i$ is replaced by $b\widehat{+}\widehat{\sum}_{i\in I} b_i.s_i$, and we can conclude as in the previous case. □.

### 5.5 Typed Timed Actions

Given an extended guard $G = (g, d)$, it can be decomposed into $G = (g \wedge \neg d, \mathit{false})\ \mathit{or}\ (d, d)$. That is, any extended guard can be expressed as the disjunction of one lazy and one eager guard. This remark motivates the definition of typed guards. If $g$ is a guard, we write $g^\lambda$ and $g^\epsilon$ to denote respectively, $g^\lambda = (g, \mathit{false})$ and $g^\epsilon = (g, g)$.

**Proposition 26.** *For $\alpha \in \{\epsilon, \lambda\}$ and a synchronization mode $g_1 | g_2 = g_1 \wedge m(g_2) \vee m(g_1) \wedge g_2$,*

- $g_1^\alpha | g_2^\alpha = (g_1 | g_2)^\alpha$
- $g_1^\epsilon \, or \, g_2^\lambda = g_1^\epsilon \, or \, (g_2 \wedge \neg g_1)^\lambda$
- $g_1^\epsilon | g_2^\lambda = (g_1 \wedge m(g_2))^\epsilon \, or \, (m(g_1) \wedge g_2)^\lambda$

*Proof.* – Let us show that $g_1^\alpha | g_2^\alpha = (g_1 | g_2)^\alpha$, for $\alpha \in \{\epsilon, \lambda\}$.

$$\begin{aligned} g_1^\epsilon | g_2^\epsilon &= (g_1 | g_2, m(g_1) \wedge g_2 \vee g_1 \wedge m(g_2)) \\ &= (g_1 | g_2, g_1 | g_2) = (g_1 | g_2)^\epsilon \\ g_1^\lambda | g_2^\lambda &= (g_1 | g_2, m(g_1) \wedge false \vee false \wedge g_2) \\ &= (g_1 | g_2, false) = (g_1 | g_2)^\lambda \end{aligned}$$

– $g_1^\epsilon \, or \, g_2^\lambda = (g_1 \vee g_2, g_1) = (g_1, g_1) or \, (g_2 \wedge \neg g_1, false) = g_1^\epsilon \, or \, (g_2 \wedge \neg g_1)^\lambda$

– By applying the definitions :

$$\begin{aligned} g_1^\epsilon | g_2^\lambda &= (m(g_1) \wedge g_2 \vee g_1 \wedge m(g_2), xm(g_1) \wedge false \vee g_1 \wedge m(g_2)) \\ &= (m(g_1) \wedge g_2, false) or \, (g_1 \wedge m(g_2), g_1 \wedge m(g_2)) \\ &= (g_1 \wedge m(g_2))^\epsilon \, or \, (m(g_1) \wedge g_2)^\lambda \end{aligned}$$

□

A consequence of the above results is that any expression built from typed guards by using synchronization modes can be reduced to an expression which is the *or* of eager and lazy guards.

It is often useful to define a type of *delayable guards* denoted by $\delta$. We take $g^\delta = g^\lambda \, or \, g\downarrow^\epsilon$, where $g\downarrow$ is the falling edge of a right-closed guard $g$.

**Proposition 27.** *Any expression involving delayable guards and the synchronization modes and, max, min, or, can be reduced into an expression which the or of delayable guards.*

$$\begin{aligned} g_1^\delta \, and \, g_2^\delta &= (g_1 \wedge g_2)^\delta \\ g_1^\delta \, max \, g_2^\delta &= (g_1 \wedge \diamond\!\!\!\!- \, g_2)^\delta \, or \, (\diamond\!\!\!\!- \, g_1 \wedge g_2)^\delta \\ g_1^\delta \, min \, g_2^\delta &= (g_1 \wedge \diamond g_2)^\delta \, or \, (\diamond g_1 \wedge g_2)^\delta \end{aligned}$$

*Proof.* We will use the properties of the falling edge operator to prove this result. Namely, $(g_1 \wedge g_2)\downarrow = g_1 \wedge g_2\downarrow \vee g_1\downarrow \wedge g_2$, $(\diamond\!\!\!\!- \, g)\downarrow = false$ and $(\diamond g)\downarrow \Rightarrow g\downarrow$.

– For *and* , we have $m(g) = g$.

$$\begin{aligned} g_1^\delta \, and \, g_2^\delta &= (g_1, g_1\downarrow) and \, (g_2, g_2\downarrow) \\ &= (g_1 \wedge g_2, g_1 \wedge g_2\downarrow \vee g_1\downarrow \wedge g_2) \\ &= (g_1 \wedge g_2, (g_1 \wedge g_2)\downarrow) \\ &= (g_1 \wedge g_2)^\delta \end{aligned}$$

– For $max$ , we have $m(g) = \diamond\!\!\!\!-\, g$.

$$\begin{aligned} g_1^\delta \, max \; g_2^\delta &= (g_1, g_1\downarrow) max \; (g_2, g_2\downarrow) \\ &= (\diamond\!\!\!\!-\, g_1 \wedge g_2 \vee g_1 \wedge \diamond\!\!\!\!-\, g_2, \diamond\!\!\!\!-\, g_1 \wedge g_2\downarrow \vee g_1\downarrow \wedge \diamond\!\!\!\!-\, g_2) \\ &= (\diamond\!\!\!\!-\, g_1 \wedge g_2, \diamond\!\!\!\!-\, g_1 \wedge g_2\downarrow) or \; (g_1 \wedge \diamond\!\!\!\!-\, g_2, g_1\downarrow \wedge \diamond\!\!\!\!-\, g_2) \\ &= (\diamond\!\!\!\!-\, g_1 \wedge g_2, (\diamond\!\!\!\!-\, g_1 \wedge g_2)\downarrow) or \; (g_1 \wedge \diamond\!\!\!\!-\, g_2, (g_1 \wedge \diamond\!\!\!\!-\, g_2)\downarrow) \\ &= (\diamond\!\!\!\!-\, g_1 \wedge g_2)^\delta or \; (g_1 \wedge \diamond\!\!\!\!-\, g_2)^\delta \end{aligned}$$

– For $min$ , we have $m(g) = \diamond g$.

$$\begin{aligned} g_1^\delta min \; g_2^\delta &= (g_1, g_1\downarrow) min \; (g_2, g_2\downarrow) \\ &= (\diamond g_1 \wedge g_2 \vee g_1 \wedge \diamond g_2, \diamond g_1 \wedge g_2\downarrow \vee g_1\downarrow \wedge \diamond g_2) \end{aligned}$$

From $(\diamond g_1)\downarrow \Rightarrow g_1\downarrow$ and $g_2 \Rightarrow \diamond g_2$, it follows that $(\diamond g_1)\downarrow \wedge g_2 \Rightarrow g_1\downarrow \wedge \diamond g_2$ and symmetrically $(\diamond g_2)\downarrow \wedge g_1 \Rightarrow g_2\downarrow \wedge \diamond g_1$. The previous equality can be rewritten :

$$\begin{aligned} g_1^\delta min \; g_2^\delta &= (\diamond g_1 \wedge g_2 \vee g_1 \wedge \diamond g_2, \\ &\quad \diamond g_1 \wedge g_2\downarrow \vee g_1\downarrow \wedge \diamond g_2 \vee (\diamond g_1)\downarrow \wedge g_2 \vee g_1 \wedge (\diamond g_2)\downarrow) \\ &= (\diamond g_1 \wedge g_2, \diamond g_1 \wedge g_2\downarrow \vee (\diamond g_1)\downarrow \wedge g_2) \\ &\quad or \; (g_1 \wedge \diamond g_2, g_1\downarrow \wedge \diamond g_2 \vee g_1 \wedge (\diamond g_2)\downarrow) \\ &= (\diamond g_1 \wedge g_2)^\delta or \; (g_1 \wedge \diamond g_2)^\delta \end{aligned}$$

□

Using typed timed actions, drastically simplifies the general model. Notice that many timed models e.g., timed Petri nets, adopt delayable semantics for their guards.

## 6 Examples

We provide two examples illustrating the use of priority choice and synchronization modes to compositionally specify systems. The first example shows how priorities can be used to achieve mutual exclusion. The second illustrates the compositional description of a traffic light controller for tramways crossing by using $min$ and $max$ synchronizations.

### 6.1 Mutual exclusion

Consider a family of periodic processes sharing in mutual exclusion a common resource. The $i$-th process has period $T_i$ and goes successively through three control states $w_i$ (wait), $e_i$ (execute), $s_i$ (sleep). We suppose that execution $e_i$ takes $E_i$ time units. A process is represented as a timed system with actions $a_i$ (awake), $p_i$ (proceed), $r_i$ (release). Two clocks $t_i$ and $x_i$ are used respectively to enforce the period and the execution time. In figure 6 we represent two such processes. The constant $D_i$ is taken $D_i = T_i - E_i$. The transition from $w_i$ to $e_i$

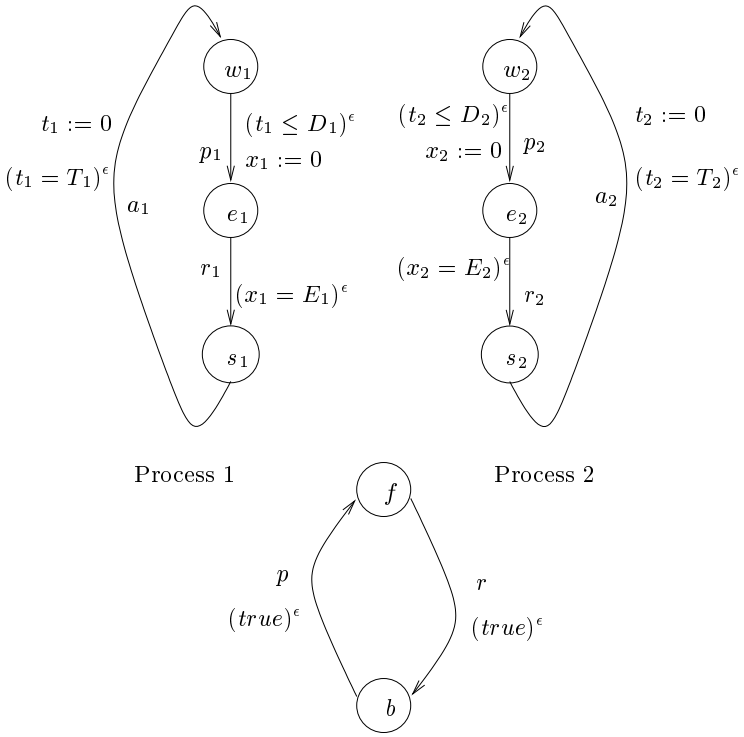
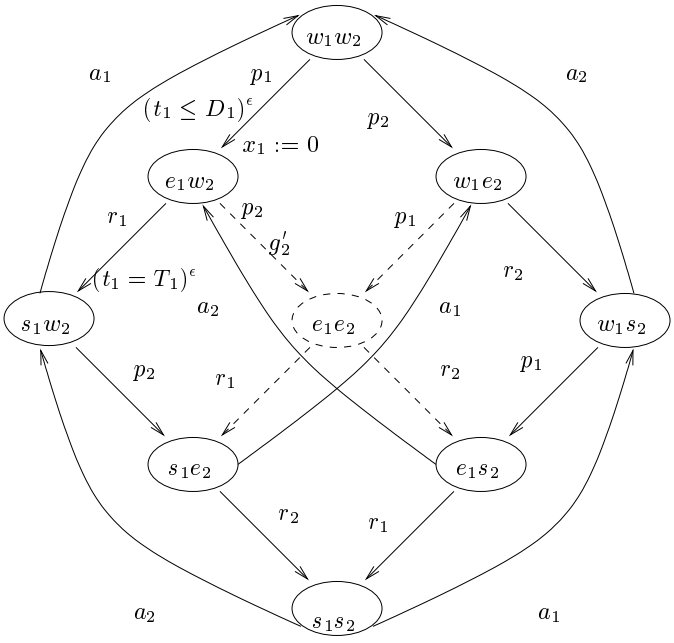



**Fig. 6.** Mutual exclusion for two processes

is taken to be eager so that no time is wasted when a component is ready to enter the critical section $e_i$.

We want to construct a scheduler guaranteeing mutual exclusion for execution. A classical solution consists in restricting the behavior of the processes by a semaphore with two actions $p$ and $r$ by taking $p_i|p \neq \bot$, $r_i|r \neq \bot$ and $\mu(p_i) = \mu(p) = \mu(r_i) = \mu(r) = and$ .

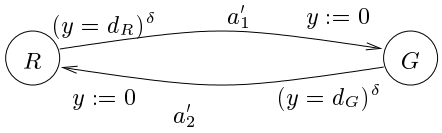
An equivalent solution can be obtained by simply imposing priorities between actions. Consider that $p_i \prec_\infty r_j$ for any pair $(i,j), i \neq j$ and take the interleaving product of the processes. It can be shown that if mutual exclusion is respected in the initial state, then it is preserved forever. Consider for instance, the interleaving product of the processes 1 and 2 under this priority restriction shown in figure 7. It is easy to check that due to priorities, the actions $p_1$ and $p_2$ will never be enabled from control states $e_1w_2$ and $w_1e_2$, respectively. Their guards will be restricted to states such that $\Box\neg(x_1 = E_1) = x_1 > E_1$ and $\Box\neg(x_2 = E_2) = x_2 > E_2$ hold respectively. It is easy to verify that for correctly initialized processes $x_i \leq E_i$ holds at control state $w_i$, which implies that transitions leading to states violating mutual exclusion will never be taken.



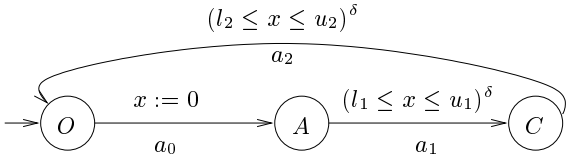

$$g_2' = ((t_1 > T_1) \wedge (t_2 \leq D_2))^\epsilon$$

**Fig. 7.** Product of process 1 and 2

### 6.2 Traffic light for tramway crossing




a. Traffic light




b. A Tramway

**Fig. 8.** Traffic light and Tramway

The light controlling the car traffic in a crossroads is a periodic timed process with two control states $G$ (Green) and $R$ (Red) and a clock $y$ to enforce sojourn times $d_G$ and $d_R$, respectively, at $G$ and $R$ (figure 8a).

We want to modify the light so as to control the traffic of tramways. When a tramway approaches the crossing, it sends a signal $a_0$ after which the light must be green within some interval $[l_1, u_1]$. This guarantees that the tramway crosses without stopping. Then, the light remains green until the tramway exits the crossing. Figure 8b represents a tramway as a process with control states $O$ (Out), $A$ (Approach), $C$ (Cross). We assume that the tramway exits the cross section within time in the interval $[l_2, u_2]$ since the beginning of the approach phase.

The modified behavior of the light can be obtained as the parallel composition of the traffic light process and of the tramway process by taking $\mu(a_1) = \mu(a_1') = min$ and $\mu(a_2) = \mu(a_2') = max$. The resulting timed controller handling one tramway (at most) is given in figure 9. It corresponds to the product of the two timed systems under the assumption of maximal progress and that all the actions interleave. The dashed transitions will never be taken due to higher priority of synchronizations. The typed guards $G_1$, $G_1'$, $G_{11}$ and $G_{22}$ are the following:

$$
\begin{aligned}
G_{11} &= (x \leq u_1 \wedge y = d_R)^\delta \vee (l_1 \leq x \leq u_1 \wedge y \leq d_R)^\delta \\
G_{22} &= (l_2 \leq x \wedge y = d_G)^\delta \vee (l_2 \leq x \leq u_2 \wedge d_G \leq y)^\delta \\
G_1 &= (l_1 \leq x \leq u_1 \wedge y > d_R)^\delta \\
G_1' &= (y = d_R \wedge x > u_1)^\delta.
\end{aligned}
$$

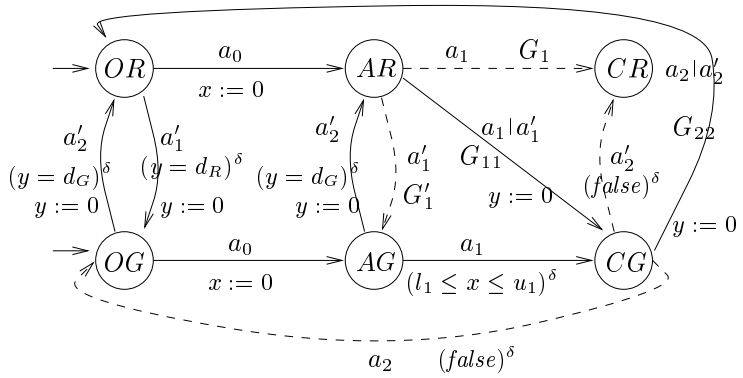



**Fig. 9.** Controller for a tramway

## 7 Discussion

The paper presents a framework for extending compositionally the description of untimed systems to timed systems by preserving time reactivity and activity of components. The adopted composition principle contrasts with the most commonly adopted which is based on strong synchronization for time progress and implies preservation of components urgency. Preserving time reactivity requires sometimes relaxing urgency constraints.

An important outcome of this work is that composition operators for untimed systems admit different timed extensions due to the possibility of controlling waiting times and "predicting" the future. The use of modalities in guards drastically increases concision in modeling and is crucial for compositionality. It does not imply extra expressive power for simple classes of timed systems, such as linear hybrid automata [ACH+95], where quantification over time in guards can be eliminated.

The definition of different synchronization modes has been motivated by the study of high level specification languages for timed systems, such as Timed Petri nets and their various extensions[SDdSS94,SDLdSS96,JLSIR97]. We have shown that the proposed framework is a basis for the study of the underlying semantics and composition techniques; if they are bounded then they can be represented as timed systems with finite control state space. Another outstanding fact is that using max-synchronization and min-synchronization, in addition to and-synchronization, drastically helps keeping the complexity of the corresponding timed system low [BST97].

The results concerning the algebraic framework itself are recent. We are currently studying their application to the compositional generation of timed models of real-time applications and in particular to scheduling.

## 8 Related Work

The problem of compositional description in languages with priorities has been principally studied for process algebras. The first work is, to our knowledge [BBK86], where is defined an untimed process algebra with a priority order on its set of actions. Later, in several papers, Cleaveland and his colleagues show the interest of priority for the specification and the verification of distributed untimed systems [CH90,CLNS96,CLN96,CLN98]. Our work is closer to the work by Insup Lee and his colleagues, [BGL97,BACC$^+$98] on the timed process algebra ACSR. The latter is a timed algebra with priorities and mutual exclusion constraints with value passing communication and dynamic priorities. It has been used for schedulability analysis of real-time systems. However, this work does not tackle compositionality issues concerning both the associativity of priority choice operators and property preservation. Another important difference is that although our priority order is static, it allows "prediction" which is essential for achieving maximal progress for timed systems.

## Acknowledgment

The authors would like to thank Yassine Lakhnech and Kim Larsen for their contribution to the improvement of this paper.

## References

[ACH$^+$95] R. Alur, C. Courcoubetis, N. Halbwachs, T. Henzinger, P. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

[BACC$^+$98] H. Ben-Abdallah, J.-Y. Choi, D. Clarke, Y.S. Kim, I. Lee, and H.-L. Xie. A process algebraic approach to the schedulability analisys of real-time systems. *Real-time Sytems, 15*, pages 189–219, 1998.

[BBK86] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Syntax and defining equations for an interrupt mechanism in process algebra. *Fundamenta Informaticae IX (2)*, pages 127–168, 1986.

[BGL97] P. Bremond-Gregoire and I. Lee. A process algebra of communicating shared resources with dense time and priorities. *Theoretical Computer Science, 189*, 1997.

[BS98] S. Bornot and J. Sifakis. On the composition of hybrid systems. In *First International Workshop Hybrid Systems : Computation and Control HSCC'98*, pages 49–63, Berkeley, March 1998. Lecture Notes in Computer Science 1386, Spinger-Verlag.

[BST97] S. Bornot, J. Sifakis, and S. Tripakis. Modeling urgency in timed systems. In *International Symposium: Compositionality - The Significant Difference*, Malente (Holstein, Germany), September 1997. Lecture Notes in Computer Science 1536, Springer Verlag.

[CH90] R. Cleaveland and M. Hennessy. Priorities in process algebra. *Information and Computation, 87(1/2)*, pages 58–77, 1990.

[CLN96] R. Cleaveland, G. Luttgen, and V. Natarajan. A process algebra with distributed priorities. In U. Montanari and V. Sassone, editors, *CONCUR '96*, pages 34–49. LNCS 1119, Springer-Verlag, August 1996.

[CLN98] R. Cleaveland, G. Luttgen, and V. Natarajan. A process algebra with distributed priorities. *Theoretical Computer Science, 195(2)*, pages 227–258, March 1998.

[CLNS96] R. Cleaveland, G. Luttgen, V. Natarajan, and S. Sims. Modeling and verifying distributed systems using priorities: A case study. *Software Concepts and Tools 17*, pages 50–62, 1996.

[HNSY94] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.

[JLSIR97] M. Jourdan, N. Layaida, L. Sabry-Ismail, and C. Roisin. An integrated authoring and presentation environment for interactive multimedia documents. In *4th Conference on Multimedia Modeling*, Singapore, November 1997. World Scientific Publishing.

[Mil83] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25:267–310, 1983.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[SDdSS94] P. Sénac, M. Diaz, and P. de Saqui-Sannes. Toward a formal specification of multimedia scenarios. *Annals of telecomunications*, 49(5-6):297–314, 1994.

[SDLdSS96] P. Sènac, M. Diaz, A. Léger, and P. de Saqui-Sannes. Modeling logical and temporal synchronization in hypermedia systems. In *Journal on Selected Areas in Communications*, volume 14. IEEE, jan. 1996.

[SY96] J. Sifakis and S. Yovine. Compositional specification of timed systems. In *13th Annual Symposium on Theoretical Aspects of Computer Science, STACS'96*, pages 347–359, Grenoble, France, February 1996. Lecture Notes in Computer Science 1046, Spinger-Verlag.